Sistemas de Big Data

Desarrolla un informe técnico detallado sobre el procesamiento de datos por lote

**Procesamiento de datos por lote**

**Implementación y análisis del procesamiento de datos por lote**

Índice

[Marco teórico 3](#_Toc185618667)

[Procesamiento de datos por lotes y procesamiento en tiempo real 3](#_Toc185618668)

[Diferencias 3](#_Toc185618669)

[Arquitectura procesamiento por lotes 3](#_Toc185618670)

[Aplicaciones y casos de uso más comunes en la industria 4](#_Toc185618671)

[Exploración de herramientas y tecnologías 4](#_Toc185618672)

[Apache Hadoop: Como funciona 4](#_Toc185618673)

[Comparación de herramientas relevantes 5](#_Toc185618674)

[Implementación práctica 6](#_Toc185618675)

[Diseña e implementa un pipeline básico de procesamiento por lotes 6](#_Toc185618676)

[Evaluación crítica 6](#_Toc185618677)

[Discute los pros y contras del procesamiento por lotes 6](#_Toc185618678)

[Bibliografía 6](#_Toc185618679)

# Marco teórico

## Procesamiento de datos por lotes y procesamiento en tiempo real

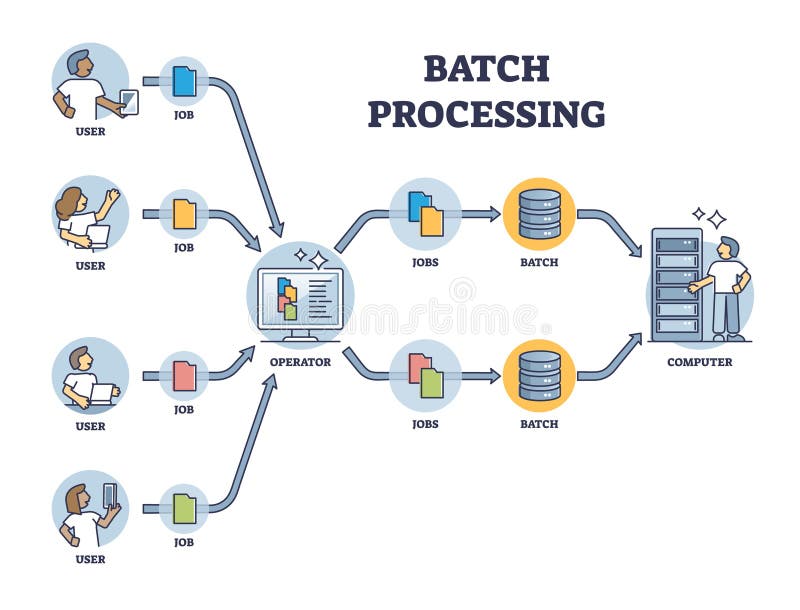
El **procesamiento por lotes** consiste en la ejecución de un programa sin supervisión del usuario, donde se almacenan datos en un único grupo o (lote) y se utilizan para trabajos de datos repetitivos y de gran volumen.

El **procesamiento en tiempo real** maneja los datos a medida que se generan. Requiere un flujo de datos constante de entrada y salida de datos.

## Diferencias

|  |  |  |
| --- | --- | --- |
| **Diferencias en cuanto a** | **Procesamiento por lotes** | **Procesamiento en tiempo real** |
| Latencia | Alta | Baja |
| Casos de uso | Análisis histórico | Monitoreo |
| Frecuencia | Por lotes en intervalos definidos | Flujo de datos continuo |
| Ejemplos | Hadoop, Spark | Apache Kafka |

## Arquitectura procesamiento por lotes



Esta sería la arquitectura del procesamiento de datos por lote, donde se realiza:

* **Ingesta:** Datos crudos desde diversas fuentes son recopilados en Amazon S3.
* **Procesamiento:** Los datos son organizados y transformados en lotes.
* **Análisis Avanzado:** Se aplican algoritmos y análisis en plataformas de Big Data.
* **Resultados:** Los datos procesados se almacenan para análisis y toma de decisiones.

## Aplicaciones y casos de uso más comunes en la industria

Las herramientas que se usan para procesamiento de datos suelen ser aquellas diseñas para manejar grandes cantidades de datos, como por ejemplo Apache Hadoop o Spark. En cuanto a almacenamiento, se podría usar HDFS o Amazon S3. Y finalmente, para visualizar se podría usar Power BI.

Como caso de uso, un ejemplo podría ser el tema de la integración de datos (ETL), donde se podría aplicar a todas las industrias, donde se extraen los datos de diversas fuentes, luego se transforman y posteriormente se cargan en los data warehouse.

Otro ejemplo puede ser el de transacciones financieras, utilizado en el ámbito de la bancos o seguros, donde se utiliza un gran número de datos.

# Exploración de herramientas y tecnologías

## Apache Hadoop: Como funciona

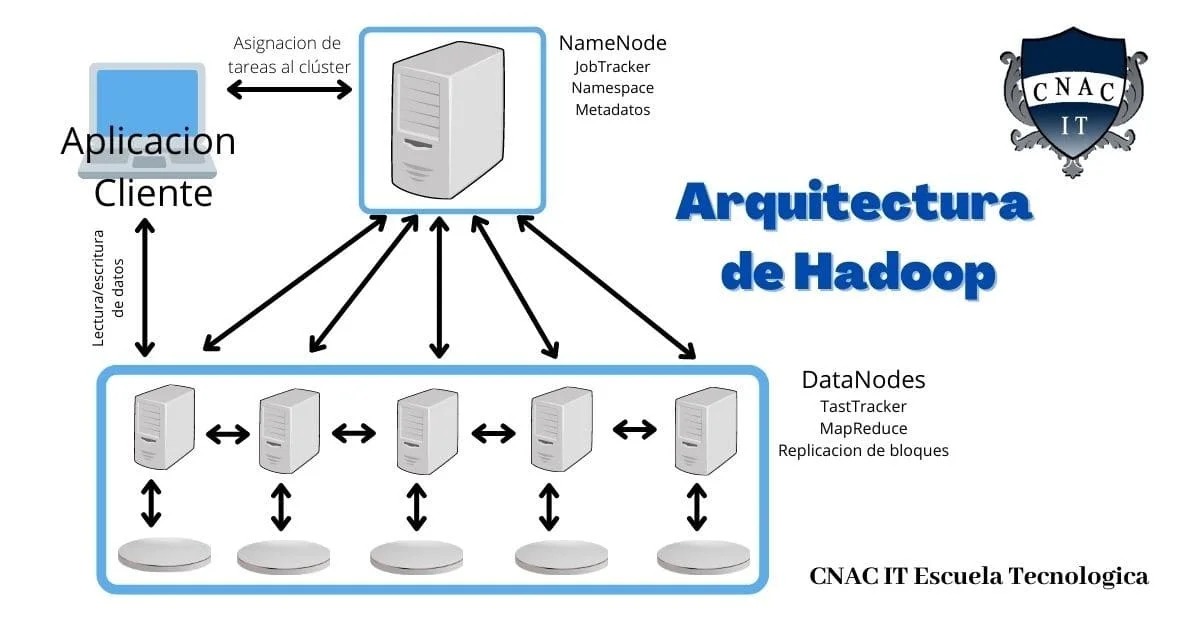
Hadoop es un marco de código abierto, que se utiliza para almacenar y procesar de manera eficiente conjuntos grandes de datos.

En lugar de utilizar una sola computadora grande para procesar y almacenar los datos, Hadoop facilita la creación de clústeres de varias computadoras para analizar conjuntos de datos masivos en paralelo y con mayor rapidez.

Existen 4 módulos principales en Hadoop:

* **HDFS**: sistema de archivos distribuido. proporciona un mejor rendimiento de datos que los sistemas de archivos tradicionales, además de una alta tolerancia a errores y compatibilidad nativa con conjuntos de datos de gran tamaño.
* **YARN**: administra y supervisa los nodos del clúster y el uso de recursos. Programa trabajos y tareas.
* **MapReduce**: Ayuda a los programas a realizar el cálculo paralelo de los datos. La tarea de Map toma los datos de entrada y los convierte en un conjunto de datos que se puede calcular en pares de valores clave. La salida de la tarea de Map la consumen las tareas de Reduce para agregar la salida y proporcionar el resultado deseado.
* **Hadoop Common**: proporciona bibliotecas Java comunes que se pueden usar en todos los módulos.

Imagen de cómo funciona:



**Datanodes**: Almacena los datos en el sistema Hadoop, y los facilita cuando son solicitados. Un clúster HDFS puede tener varios DataNode, con datos replicados entre ellos.

**Namenode**: Administra el espacio de nombres del sistema de archivos.

Como **características principales**, Apache Hadoop destaca por:

* **Escalabilidad**: Se pueden ir añadiendo más nodos al clúster
* **Tolerancia a fallos**: Replica datos en múltiples nodos
* **Alto rendimiento**: Está optimizado para trabajar con grandes volúmenes de datos
* **Ecosistema robusto**: Integra herramientas para expandir sus capacidades

## Comparación de herramientas relevantes

|  |  |  |
| --- | --- | --- |
| **Característica** | **Apache Hadoop** | **Apache Spark** |
| Modelo de procesamiento | Procesamiento en disco (MapReduce) | Procesamiento en memoria |
| Velocidad | Mas lento debido al uso de disco | Mas rápido por uso de memoria |
| Facilidad de uso | Es más complejo, utiliza MapReduce | Soporta APIs |
| Casos de uso | Procesamiento de grandes lotes de datos | Procesamiento en tiempo real y lotes |
| Gestión de recursos | Se basa en YARN | Puede o no usar YARN |

# Implementación práctica

## Diseña e implementa un pipeline básico de procesamiento por lotes

Primero seleccionamos un dataset, yo elegí [Cryptocurrency Historical Prices](https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory)

Después vamos a la página de [Apache Hadoop](https://hadoop.apache.org/) y lo descargamos.

No sé implementar el pipeline, tampoco vi apuntes de ello. Traté de investigar y no lo conseguí sacarlo.

# Evaluación crítica

## Discute los pros y contras del procesamiento por lotes

Hadoop es bueno para proyectos donde se requiere usar grandes cantidades de datos, en un entorno distribuido, por ejemplo, para temas de análisis o ETL.

Como pros, la alta fiabilidad de datos, y la escalabilidad creo que es en lo que más destaca, y los contras la falta de flexibilidad frente a cambios, latencia/lentitud por el procesamiento en disco.

Si tu proyecto incluye procesamiento en tiempo real, mejor usar Apache Spark, ya que está mas preparado para ese tipo de entornos.

# Bibliografía

<https://www.jvs-informatica.com/blog/glosario/procesamiento-por-lotes-batch/#:~:text=Qu%C3%A9%20es%20el%20procesamiento%20por%20lotes&text=El%20procesamiento%20por%20lotes%2C%20tambi%C3%A9n,el%20usuario%20realice%20ninguna%20interacci%C3%B3n>.

<https://www.astera.com/es/type/blog/batch-processing-vs-stream-processing/#:~:text=El%20procesamiento%20por%20lotes%20recopila,real%20tal%20como%20se%20reciben.&text=El%20procesamiento%20por%20lotes%20suele,maneja%20grandes%20vol%C3%BAmenes%20de%20datos>.

<https://aws.amazon.com/es/what-is/batch-processing/>

<https://www.profesionalreview.com/2018/11/25/que-es-el-procesamiento-batch/>

<https://www.datacamp.com/es/blog/batch-vs-stream-processing>

<https://hadoop.apache.org/>

<https://aws.amazon.com/es/what-is/hadoop/>

<https://es.wikipedia.org/wiki/Apache_Hadoop>

<https://www.ibm.com/es-es/topics/hadoop>

<https://www.cnac.es/noticias/que-es-hadoop/>

<https://aws.amazon.com/es/compare/the-difference-between-hadoop-vs-spark/#:~:text=Apache%20Hadoop%20allows%20you%20to,against%20data%20of%20any%20size>.

<https://www.ibm.com/think/insights/hadoop-vs-spark>

<https://databay.solutions/hadoop-vs-spark-comparacion-de-tecnologias-de-big-data/>

<https://cloud.google.com/learn/what-is-apache-spark?hl=es#:~:text=Muchas%20empresas%20usan%20Spark%20para,tanto%20estructurados%20como%20sin%20estructurar>.